

Reinforcement Learning: An Introduction

Summary Notes

Mrinank Sharma

March 28, 2020

1 Introduction

Other than the **agent** and the **environment**, there are four main subelements of an RL system:

Elements of Reinforcement Learning

1. The *policy* maps perceived states of the environment to actions. This could be a lookup table or a search process.
2. The *reward signal* defines the goal of a reinforcement learning problem. The environment sends rewards to an agent that has the sole objective of maximising this reward.
3. The *value function* specifies what is good in the long run. The *value* of a state is the total amount of reward an agent can expect to accumulate over the future starting from that state.
4. Some RL systems have *models* of the environment that can mimic the behaviour of the environment. Models are used for planning.

Whilst rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account which states are likely to follow. We choose actions based on values, even though values are derived from rewards.

Rewards and Value Functions

It is possible to learn policies without estimating value functions by using evolutionary methods. However, they tend to ignore much of the useful structure and are not well suited for RL tasks as they ignore relevant information, such as the states which are passed and the actions which are selected.

Evolutionary Methods

2 Multi-armed Bandits

*The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that **evaluates** the actions taken rather than **instructs** by giving correct actions.*

This form of feedback creates the need for active exploration.

Definition 2.1 (*k*-armed Bandit) *In the *k*-armed bandit problem, you are repeatedly faced with a choice among *k* different actions. After each choice, a numerical reward is received from a **stationary** probability distribution associated with the selected action. The objective is to maximise the expected total reward over some time period.*

k-armed bandits

The value of action *a* is the expected reward given *a* was chosen:

Value

$$q_*(a) = \mathbb{E}[R_t | A_t = a]. \quad (1)$$

If we knew the values, problem solved - just pick the action with the maximum value. However, we do not know these. Denote the *estimated* value as $Q_t(a)$. A simple way to estimate the action-value is a *sample average*, which can be calculated incrementally.

Note that purely greedy methods are best if there is no noise in the reward signal. With ϵ -greedy algorithms, the best value of ϵ will depend on the amount of noise, which

Reward Variance

determines the number of samples for the sample average to converge. However, if the problem is *non-stationary*, we still need to explore.

If the bandit is non-stationary, we might use the following rule for value estimates:

Non-stationary Problems

$$Q_{n+1}(a) = Q_n(a) + \alpha_n(a) [R_n - Q_n(a)]. \quad (2)$$

It is known that convergence with probability 1 happens if the following conditions hold:

Convergence Guarantees

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty, \text{ and } \sum_{n=1}^{\infty} \alpha_n(a)^2 < \infty. \quad (3)$$

The first condition ensures the steps are sufficiently large to overcome an initial condition and the second condition ensures that the steps become small enough to give convergence. However, step sizes which meet these conditions are rarely used in applications because they seem to converge slowly or require considerable tuning.

With these techniques, the initialisation effectively becomes an additional parameter which needs to be chosen. They can be used to provide prior knowledge and can encourage exploration, for example, by being optimistic. However, this sort of exploration is not useful for non-stationary problems (the task changes creates a renewed need for exploration).

The intuition behind UCB is that it's better, when exploring, to select among the non-greedy actions which have the *potential* for being optimal. To do this, we maximise an estimate of the arm's value summed with some sort of uncertainty estimate. UCB often performs well but can be difficult to extend to RL from bandits.

UCB

We don't have to use value based methods. Instead, we could directly learn a preference over the actions and modify our preferences to give us higher rewards e.g., increasing our preference for an action if selecting it leads to a better outcome than expected.

Definition 2.2 (Contextual Bandits) *In a contextual bandit, the agent is faced with a series of multi-armed bandit problems, each associated with some information i.e., the agent has information about which bandit problem is being faced at any timestep.*

Contextual Bandits

Contextual bandits are intermediate between multi-armed bandits and the full RL problem; they involve learning a policy (i.e., a mapping from state information to actions) but similar to k -armed bandits, **the action selected influences only the immediate reward and not the next situation**. If we drop this constraint, we have the full RL problem.

3 Finite Markov Decision Processes

Definition 3.1 (Finite Markov Decision Process (MDP)) *A finite Markov Decision Process (MDP) is made up of:*

MDP

- A sequence of discrete time steps, $t = 0, 1, 2, \dots$
- A discrete set of states, $S_t \in S$.
- A set of actions which the agent can take, $A_t \in \mathcal{A}(S_t)$.
- A finite set of numerical rewards, $R_t \in \mathcal{R} \subset \mathbb{R}$. In this case, there is a well defined discrete transition distribution:

$$p(s', r | s, a) \triangleq \Pr[S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a], \quad (4)$$

which defines the **dynamics** of the MDP.

Note the Markov assumption; probabilities over the next state and reward depend only on the current state and action. This can be interpreted as a restriction on the state - the state must include information about all aspects of the past agent-environment interaction which will affect the future.

Interpreting the Markov Assumption

Given p , we can compute the following functions of interest:

$$p(s' | s, a) \triangleq \Pr[S_t = s' | S_{t-1} = s, A_{t-1} = a], \quad (5)$$

$$r(s, a) \triangleq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a], \quad (6)$$

and

$$r(s, a, s') \triangleq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s']. \quad (7)$$

A general rule of thumb is that anything that cannot be changed *arbitrarily* by the agent is considered to be outside of it and thus part of the environment. The agent-environment boundary represents the limit of the agent's absolute control, not of its knowledge.

Agent vs Environment

The goal of the agent is formalised in terms of the reward signal which passes from the environment to the agent. This is one of the most distinctive features of RL.

All of what we mean by goals and purposes can be well thought of as the maximisation of the expected value of the cumulative sum of a received scalar signal.

Reward Hypothesis

The reward signal is not the place to impart to the agent prior knowledge about *how* to achieve a task; if we do this, the agent may only achieve subgoals without actually achieving the real goal. A better place to impart this knowledge is the initial policy or value function. **How does this link to reward shaping?**

Episodic tasks terminate at some timestep, T , whilst continuing tasks do not. In the first case, we simply use the (expected) sum of rewards whilst we introduce discounting in the infinite horizon case i.e.,

Discount Rates

$$G_t \triangleq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \quad (8)$$

G_t is the *discounted return*. The closer the value of γ is to 1, the more farsighted the agent is.

We convert episodic tasks to continuing tasks by introducing a special *absorbing state* which transitions only to itself and returning zero reward. We can write:

Unifying Notation

$$G_t \triangleq \sum_{k=t+1}^T \gamma^{k-t-1} R_k, \quad (9)$$

allowing for $T = \infty$ and $\gamma = 1$. **The book says not both - but I think that should be allowed for an episodic task with absorbing state.**

Definition 3.2 (Policy) A policy is a mapping from states to probabilities of selecting each possible action. $\pi(a|s)$ is the distribution over actions, A_t , given the current state, S_t .

Policy

Definition 3.3 (Value Function) The value function of state s under policy π is the expected return when starting in s and following π .

Value Function

$$v_{\pi}(s) \triangleq \mathbb{E}_{\pi}[G_t | S_t = s] \quad (10)$$

Definition 3.4 (Action-Value Function) The action-value function of action a , state s under policy π is the expected return when starting in s and following π after first performing a .

Action-Value Function

$$q_{\pi}(s, a) \triangleq \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \quad (11)$$

Value functions satisfy recursive relationships.

A policy, π , is better than policy π' iff $v_{\pi}(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$ with a strict inequality for at least one state. Optimal policies are better than or equal to all other policies. There may be multiple optimal policies, but they all share the same state-value function,

Optimal Policies

$$v_*(s) = \max_{\pi} v_{\pi}(s), \text{ for all } s \in \mathcal{S}. \quad (12)$$

The optimal action-value function is defined analogously.

The Bellman optimality equation for the value function is:

Bellman Optimality Equality

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s', r \in \mathcal{R}} p(s', r | s, a) [r + \gamma v_*(s')], \quad (13)$$

which expresses the intuition that the optimal value of a state is equal to the expected return of the best action from that state. Similarly, for the action-value function,

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a')]. \quad (14)$$

For a finite MDP, the Bellman optimality equations give a system of equations which can be solved for $v_*(s)$. Once we have the optimal value functions, it is easy to determine an optimal policy by acting greedily (or performing a one-step search). **If you use v_* to select actions in the short-term, the greedy policy is optimal in the long term since this value function takes into account reward consequences.** The optimal expected long-term return is locally and immediately available for each state so a one-step lookahead search yields optimal actions.

Note that the action-value function effectively caches the results of all of the one-step lookahead searches. This function can be maximised (with respect to the action) to find the optimal policy.

This all sounds great. However, we run into the following problems:

Caveats

- We need to accurately know the dynamics of the environment.
- We need to have enough computational resources.
- The Markov property needs to hold.